# Отчет по домашнему заданию № 9 – Балансировка и отказоустойчивость

В результате выполнения ДЗ было уменьшено количество точек отказа в тестовом приложении и тренировались навыки:

* проектирование отказоустойчивых архитектур;
* настройка nginx;
* настройка HAProxy

В процессе выполнения ДЗ был создан проект, в котором с помощью docker-compose запускалось несколько сервисов с необходимыми настройками конфигурации.

Структура проекта:

.

├── docker-compose.yml

├── master

│ ├── init-master.sh

│ └── postgresql.conf

├── slave

│ └── init-slave.sh

├── haproxy.cfg

├── nginx.conf

└── app

├── Dockerfile

└── app.py

Где:

Docker-compose.yml

x-postgres-common:

  &postgres-common

  image: postgres:14-alpine

  user: postgres

  restart: always

  healthcheck:

    test: 'pg\_isready -U postgres --dbname=postgres'

    interval: 10s

    timeout: 5s

    retries: 5

services:

  postgres-master:

    <<: \*postgres-common

    container\_name: pg-master

    ports:

      - 54321:5432

    environment:

      POSTGRES\_USER: postgres

      POSTGRES\_DB: postgres

      POSTGRES\_PASSWORD: pass

      POSTGRES\_HOST\_AUTH\_METHOD: "scram-sha-256\nhost replication all 0.0.0.0/0 md5"

      POSTGRES\_INITDB\_ARGS: "--auth-host=scram-sha-256"

    command: |

      postgres

      -c wal\_level=replica

      -c hot\_standby=on

      -c max\_wal\_senders=10

      -c max\_replication\_slots=10

      -c hot\_standby\_feedback=on

    volumes:

      - master\_data:/var/lib/postgresql/data

      - ./00\_init.sql:/docker-entrypoint-initdb.d/00\_init.sql

    networks:

      - appnet

  postgres-slave1:

    <<: \*postgres-common

    container\_name: pg-slave1

    ports:

      - 54322:5432

    environment:

      PGUSER: rep1

      PGPASSWORD: passrep1

    command: |

      bash -c "

      until pg\_basebackup --pgdata=/var/lib/postgresql/data -R --slot=replication\_slot1 --host=postgres-master --port=5432

      do

      echo 'Waiting for primary to connect...'

      sleep 1s

      done

      echo 'Backup done, starting replica...'

      chmod 0700 /var/lib/postgresql/data

      postgres

      "

    depends\_on:

      - postgres-master

    volumes:

      - slave1\_data:/var/lib/postgresql/data

    networks:

      - appnet

  postgres-slave2:

    <<: \*postgres-common

    container\_name: pg-slave2

    ports:

      - 54323:5432

    environment:

      PGUSER: rep2

      PGPASSWORD: passrep2

    command: |

      bash -c "

      until pg\_basebackup --pgdata=/var/lib/postgresql/data -R --slot=replication\_slot2 --host=postgres-master --port=5432

      do

      echo 'Waiting for primary to connect...'

      sleep 1s

      done

      echo 'Backup done, starting replica...'

      chmod 0700 /var/lib/postgresql/data

      postgres

      "

    depends\_on:

      - postgres-master

    volumes:

      - slave2\_data:/var/lib/postgresql/data

    networks:

      - appnet

  haproxy:

    image: haproxy:2.9

    container\_name: haproxy

    ports:

      - "5433:5433"

    volumes:

      - ./haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg:ro

    depends\_on:

      - postgres-slave1

      - postgres-slave2

    networks:

      - appnet

  backend1:

    build: ./app

    container\_name: backend1

    ports:

      - "8001:8000"

    networks:

      - appnet

  backend2:

    build: ./app

    container\_name: backend2

    ports:

      - "8002:8000"

    networks:

      - appnet

  nginx:

    image: nginx:1.25

    container\_name: nginx

    ports:

      - "8003:80"

    volumes:

      - ./nginx.conf:/etc/nginx/conf.d/default.conf:ro

    depends\_on:

      - backend1

      - backend2

    networks:

      - appnet

volumes:

  master\_data:

  slave1\_data:

  slave2\_data:

networks:

  appnet:

    driver: bridge

master/init-master.sh

#!/bin/bash

set -e

# Создаём пользователя для репликации

psql -U "$POSTGRES\_USER" -d postgres <<-EOSQL

    CREATE ROLE $REPL\_USER REPLICATION LOGIN PASSWORD '$REPL\_PASSWORD';

EOSQL

master/postgresql.conf

listen\_addresses = '\*'

wal\_level = replica

max\_wal\_senders = 10

wal\_keep\_size = 64

archive\_mode = off

hot\_standby = on

slave/init-slave.sh

#!/bin/bash

set -e

# Если данных нет — делаем basebackup с мастера

if [ -z "$(ls -A $PGDATA)" ]; then

    echo "Running pg\_basebackup from $MASTER\_HOST..."

    rm -rf $PGDATA/\*

    PGPASSWORD=$REPL\_PASSWORD pg\_basebackup \

        -h $MASTER\_HOST \

        -U $REPL\_USER \

        -D $PGDATA \

        -Fp -Xs -P -R \

        -d "postgres"

    chmod 0700 $PGDATA

fi

# Запускаем стандартный entrypoint Postgres

exec docker-entrypoint.sh postgres

haproxy.cfg

global

    log stdout format raw local0

    maxconn 4096

    daemon

defaults

    log global

    mode tcp

    option tcplog

    retries 3

    timeout connect 5s

    timeout client 1m

    timeout server 1m

frontend psql\_read

    bind \*:5433

    default\_backend psql\_slaves

backend psql\_slaves

    balance roundrobin

    option tcp-check

    server slave1 postgres-slave1:5432 check

    server slave2 postgres-slave2:5432 check

nginx.conf

upstream backend\_pool {

    server backend1:8000 max\_fails=3 fail\_timeout=30s;

    server backend2:8000 max\_fails=3 fail\_timeout=30s;

}

server {

    listen 80;

    location / {

        proxy\_pass http://backend\_pool;

        proxy\_set\_header Host $host;

        proxy\_set\_header X-Real-IP $remote\_addr;

    }

}

app/Dockerfile

FROM python:3.11-slim

WORKDIR /app

COPY app.py /app

RUN pip install flask psycopg2-binary

CMD ["python", "app.py"]

app/app.py

from flask import Flask

import os

import psycopg2

app = Flask(\_\_name\_\_)

@app.route("/")

def index():

    try:

        conn = psycopg2.connect(

            host="haproxy",

            port=5433,

            dbname="postgres",

            user="postgres",

            password="pass"

        )

        cur = conn.cursor()

        cur.execute("SELECT NOW();")

        result = cur.fetchone()

        time\_now = result[0]

        cur.execute("SELECT inet\_server\_addr();") # inet\_server\_addr()  inet\_server\_port()

        result = cur.fetchone()

        addr\_now = result[0]

        cur.close()

        conn.close()

        return f"Hello from {os.getenv('HOSTNAME')}! DB address: {addr\_now}, DB Time: {time\_now}"

    except Exception as e:

        return f"DB connection error: {e}"

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(host="0.0.0.0", port=8000)

Тестовое приложение просто выводит HOSTNAME процесса backend, адрес слейва PostgreSQL и текущую дату для контроля изменений.

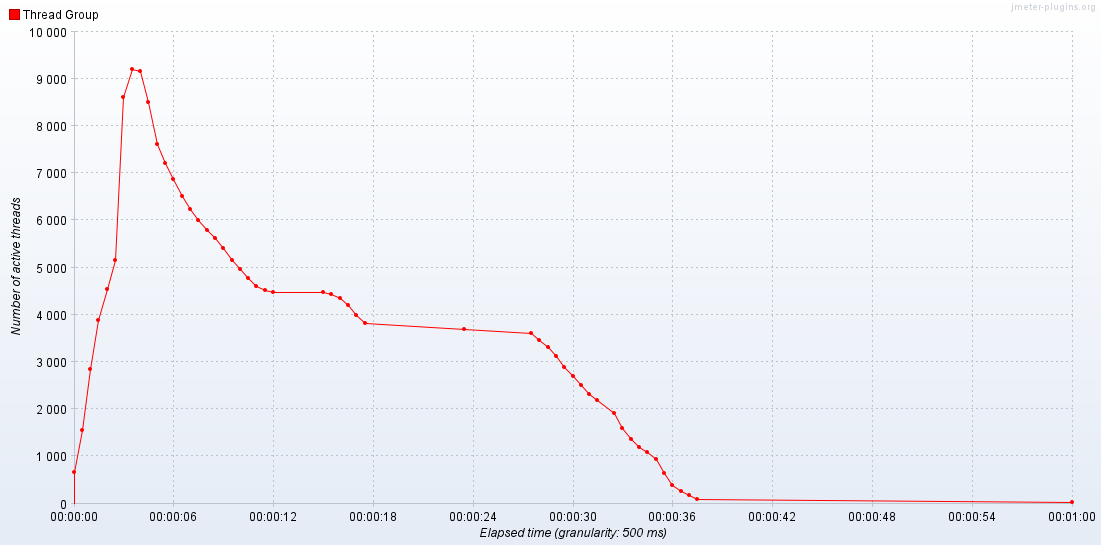
При последовательных запросах <http://localhost:8003> выводятся ответы из разных сервисов backend и базы данных, которые регулируются сервисами nginx и haproxy.

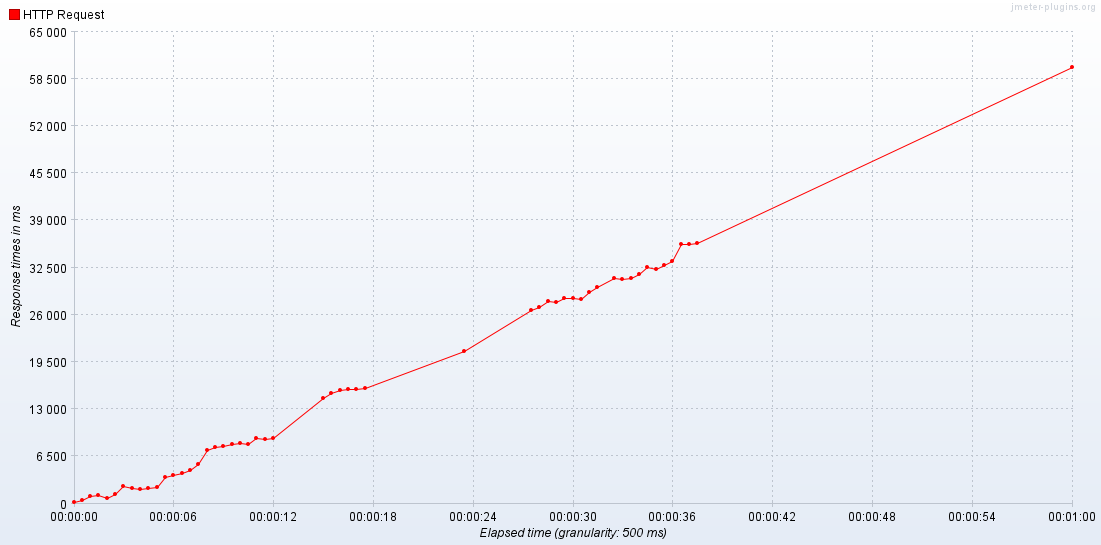
Нагрузка создавалась с помощью программы JMeter, проект нагрузочного теста приложен в файле TestLoad.jmx

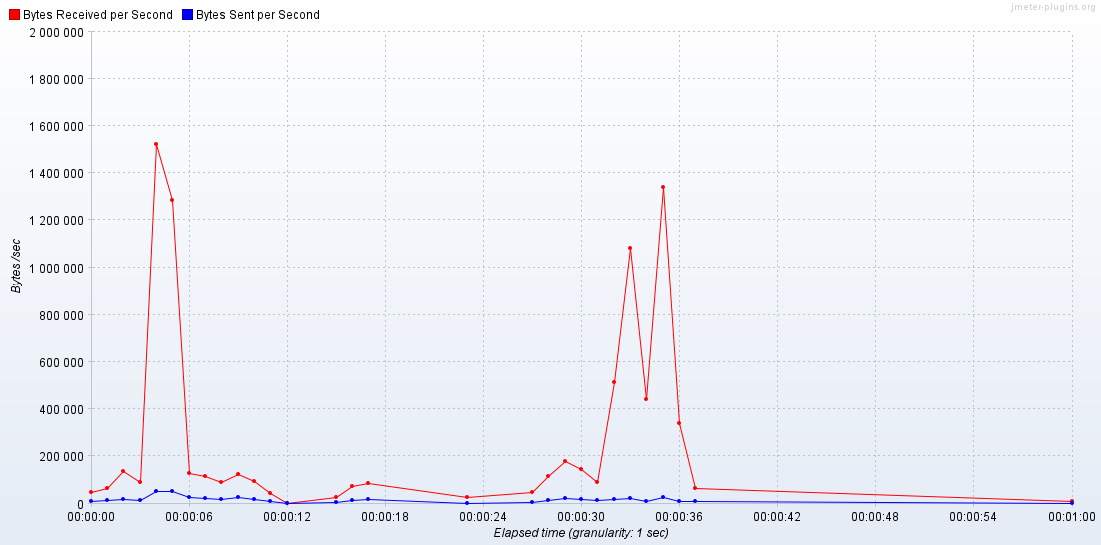
Количество пользователей задано 10000 с интервалом нарастания по 1 секунде. Под нагрузкой последовательно останавливались сервисы одного из слейвов PostgreSQL и один инстанс backend.

По результатам тестирования в файде results.csv и графикам видно, что во время выключения сервисов работоспособность системы не терялась, хотя не все запросы приходили с ответом 200. Это значит, что отказоустойчивость с помощью балансировщиков соблюдалась.

Графики нагрузочного тестирования:

ActiveThreads

ResponseTimes

BytesThoughput

ResponseLatencies